On Continuous-Action Q-Learning via Tile Coding Function Approximation

Alexander A. Sherstov and Peter Stone
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712 USA
{sherstov, pstone}@cs.utexas.edu

Abstract

Reinforcement learning (RL) is a powerful machine-learning methodology that has an established theoretical foundation and has proven effective in a variety of small, simulated domains. There has been considerable work on applying RL, a method originally conceived for discrete state-action spaces, to problems with continuous states. The extension of RL to allow continuous actions, on the other hand, has seen relatively little research. One proposed approach to allowing continuous actions is to represent the value function using a tile-coding function approximator. We introduce a simulated domain for the controlled study of this method in conjunction with Q-learning and report empirical results on its performance under different parameterizations. Our experimental findings contribute a deeper understanding of the workings of tile coding in continuous-action domains, provide guidance to parameter choices, and point out an improvement on this method which we verify empirically.

1 Introduction

Reinforcement learning (RL) is a powerful machine-learning methodology that has an established theoretical foundation and has proven effective in a variety of small, simulated domains. The application of RL, a method originally conceived for discrete state-action spaces, to practical tasks has been complicated by the ubiquity of continuous variables. There has been much research into extending RL to continuous-state, discrete-action domains. Examples include backgammon [8], soccer keepaway [5] as well as the acrobot, mountain car, and puddle world simulated domains [7]. Continuous-action domains, on the other hand, have been explored to a very limited extent. A few methods are reported in [1, 2, 4].

One proposed approach to allowing continuous actions is to represent the value function using a *tile coding* function approximator. This approach was successful on the double integrator and pendulum swing-up tasks [4]. However, we are not aware of any subsequent, more detailed analysis of this method. We introduce a simulated domain for the controlled study of this method in conjunction with Q-learning [10] and report empirical results on its performance under different parameterizations. Our experimental findings contribute a deeper understanding of the workings of tile coding in continuous-action domains, provide guidance to parameter choices, and point out an improvement on this method which we verify empirically.

The rest of this paper is organized as follows. Section 2 provides a brief overview of RL and describes tile coding and our testbed domain. Experimental results and their interpretation are presented in Sections 3 and 4, respectively. Section 5 builds on our findings to propose an improvement on the original value estimation approach. Section 6 concludes with a summary of our results.

2 Background

This section introduces reinforcement learning (RL) and tile coding and describes the testbed domain used in our experiments.

2.1 Reinforcement learning

In RL [6], a *learner* is placed in a poorly understood, possibly stochastic and non-stationary *environment*. The learner interacts with the environment at discrete time steps. At every time step, the learner can observe and change the environment's *state* through its *actions*. In addition to state changes, the environment responds to the learner's actions with a *reward*, a scalar quantity that represents the immediate utility of taking a given action in a given state. The learner's objective is to develop a *policy* (a mapping from environment states to actions) that maximizes its long-term return.

More formally, the prototypical RL problem is given by the quadruple $\{\mathcal{S}, \mathcal{A}, T, R\}$, where $\mathcal{S}$ is a finite set of states; $\mathcal{A}$ is a finite set of actions; $T : S \times A \times S \to [0, 1]$ is a *transition function* that specifies the probability of observing a certain state after taking a given action in a given state; and $R : S \times A \to \mathbb{R}$ is a *reward function* that specifies the expected reward upon taking a given action in a given state. The learner refines its policy by iteratively updating a *value function* $\hat{Q} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that approximates the expected long-term return obtained by taking a certain action in a given state and acting "optimally" thenceforth. Given a stream of rewards $r_0, r_1, r_2, \ldots$, the associated return is defined as $\sum_{i=0}^{\infty} \gamma^i r_i$, where $0 \leq \gamma \leq 1$ is the *discount factor*.

2.2 Tile coding

In practical applications of RL, states and actions are defined by continuous parameters such as distances and voltages. As a result, the sets $\mathcal{S}$ and $\mathcal{A}$ are infinite, and learning the value function requires some form of *function approximation*. To this end, we use *tile coding* [6], a function-approximation technique successfully employed in numerous reinforcement-learning systems [3, 4, 5, 7, 9]. In tile coding, the continuous variable space is partitioned into *tiles*. Any such partition is called a *tiling*. The method uses several overlapping tilings and for each tiling, maintains the weights of its tiles. The approximate value of a given point is found by summing the weights of the tiles, one per tiling, in which it is contained. Given a training example, the method adjusts the weights of the involved tiles by the same amount to reduce the error on the example.

Fig. 1 illustrates tile coding as it is used in this paper. The variable space consists of a single continuous variable x. The tiles are all the same width and adjacent tilings are offset by the same amount, the type of tiling organization we refer to as *canonical*. Fig. 1 also illustrates the computation of value estimates. A tiling organization such as those in Fig. 1 is given by tile width w and the number of tilings t. The ratio w/t is the *resolution* of the tiling organization. Speaking of tiling organizations that provide the same resolution, we refer to the number of tilings as the *breadth of generalization* since tiling organizations with more tilings generalize more broadly. A degenerate form of tile coding is a single-tiling organization, which does not generalize across tile boundaries (the rightmost organization in Fig. 1). An initial result regarding canonical tiling organizations in univariate spaces is (a proof sketch is in the appendix):

Theorem 1 *For every $s, t \geq 1$, the sets of functions representable by s- and t-tiling canonical univariate organizations with the same resolution are identical.*

For example, the three organizations in Fig. 1 are functionally equivalent. Despite this representational equivalence, tiling organizations with more tilings generalize more broadly and perform differently on RL tasks. The purpose of this paper is to identify how varying the degree of generalization—while preserving representational equivalence—affects performance.

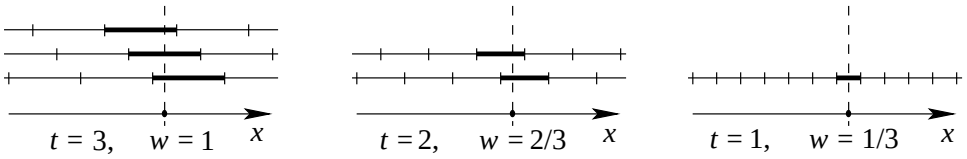


Figure 1: Three-, two-, and one-tiling canonical organizations providing the same resolution $r = 1/3$. The number t of tilings and tile width w are specified for each organization. In each case, the weights of the highlighted tiles are summed to obtain the value estimate for the indicated point.

2.3 Testbed domain

Since our aim is to study function approximation in continuous action domains, we introduce a testbed domain that has discrete states and continuous actions. This domain, shown along with an optimal policy in Fig. 2, is a 10×4 grid world. Two locations of the grid world are designated as "start" and "goal," with the learner's objective being to navigate from the start cell to the goal cell. Another type of cell is a wall that the learner cannot pass through. Finally, certain cells are designated as an abyss. This grid world task is episodic, ending with the learner falling into the abyss ("stepping off the cliff") or successfully entering the goal state. The (discrete) state variables observable to the learner are the cell coordinates x and y. The actions at the learner's disposal are of the form (d, p), where d is one of the four cardinal directions and p is a real-valued number between 0 and 1. The learner moves in the requested direction with probability $F(x, y, p)$, and in one of the three other directions with probability $(1 - F(x, y, p))/3$. Moves into walls and off the edge of the grid world result in no change of cell. F is a cell-dependent function that maps p to $[0.5, 1]$. The two "extreme" F functions are shown in Fig. 3, and the F functions for all other cells are successive interpolations between these two.[1] This design of F was intended to ensure continuity as well as multiple local maxima and minima.

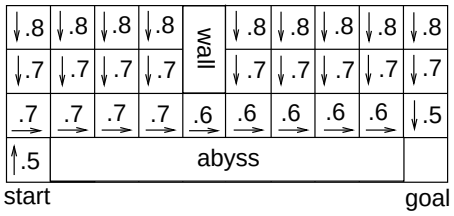


Figure 2: The grid world map and optimal policy.

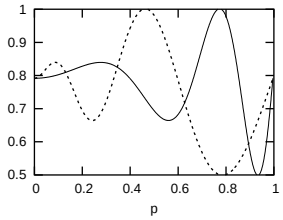


Figure 3: The two extreme $F(p)$ functions.

The learner initially has no information regarding T, R, or any of the F's. Thus, the challenge is to identify at each cell the right direction of travel and the p value that maximizes the probability of this move. We use tile coding in the p variable to approximate the value function for every distinct setting of (x, y, d). Our experiments use two different reward

[1]The exact functional form is $F(x, y, p) = \frac{1}{3} w(p, x, y) \sin(4\pi w(x, y, p)) + \frac{19}{24}$, where $w(\cdot)$ is a warping function that applies a different monotonic transformation of the p range for each cell. As a result, the optimum p value is different for every cell. As the cells are traversed in row-major order, the F curve gradually transforms from one extreme in Fig. 3 to the other.

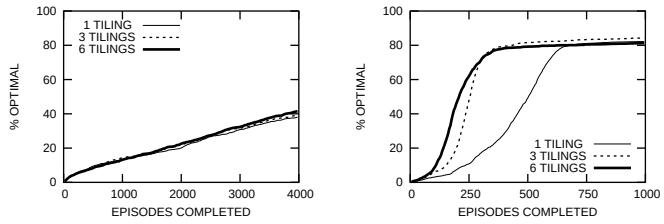
functions to guide the learner to the goal (see Section 3). The optimal policy (directions and rounded p values) is shown in Fig. 2.

3 Initial empirical results

In this section, we present empirical results in three specific scenarios illustrating the effects of the breadth of generalization on performance. The settings of generalization breadth compared are 1, 3, and 6 tilings, all reasonable choices given the target function curves in Fig. 3. Resolution was fixed at 0.04, corresponding to 26, 10, and 6 tiles per tiling in the 1, 3, and 6 tiling cases, respectively.[2] All experiments in this paper used Q-learning with ϵ-greedy action selection. The parameter settings were: $\alpha = 0.1$, $\gamma = 0.99$, $\epsilon = 0.05$, and $\lambda = 0$, except where indicated otherwise. The experiments used two reward functions, an "informative" one with -1 assigned on every nonterminal transition, -100 on stepping off the cliff, and $+100$ on reaching the goal cell; and another, "uninformative" reward function with zero reward assigned on all transitions except the one to the goal cell $(+100)$. Because of the discount parameter $\gamma = 0.99 < 1$, the optimal policy is the same under both functions. The $\hat{Q}$ estimates were initialized to 0 on all runs. The metric in all experiments was the value of the start state under the best policy discovered so far (as a fraction of optimal), as determined by an external policy-evaluation module. This model-based evaluation module was unrelated to the model-free algorithm used to learn the policies. Every performance curve in the graphs represents the episode-wise average of 250 independent runs with all identical settings.

3.1 Early performance boost due to generalization

Fig. 4 plots early performance obtained using the uninformative reward function (*a*) and the informative one (*b*). The graphs show a performance boost due to generalization when the informative reward function is used, but no observable differences with the uninformative reward function.



a *b*

Figure 4: Early performance with the uninformative (*a*) and informative (*b*) reward functions. The ordering of the curves in *b* is statistically significant at confidence level 0.005 between episodes 92 and 280.

3.2 Small step size α: long-term performance boost due to generalization

Fig. 5 plots performance over the first 50000 episodes, a substantial allotment of learning time. The reward function used is the uninformative one, chosen to control for the apparent advantage enjoyed by broad-generalizing learners in Section 3.1.[3] At $\alpha = 0.5$, there is no observed benefit to generalization. As α is decreased to 0.10 and then to 0.05, however, generalization becomes increasingly beneficial.

[2] Without offsetting, the tile counts would have been $\lceil \frac{1}{1 \cdot 0.04} \rceil$=25, $\lceil \frac{1}{3 \cdot 0.04} \rceil$=9, and $\lceil \frac{1}{6 \cdot 0.04} \rceil$=5. The extra tile is needed because with offsetting, there are partial tiles at either end of the p range.

[3] As the analysis in Section 4 will show, the results observed in Sections 3.1 and 3.2 are due to different causes which this experimental setup serves to isolate.

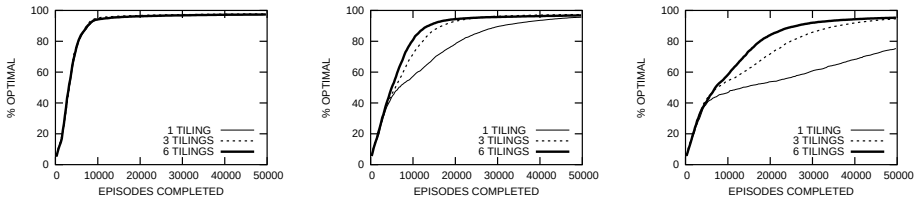


Figure 5: Performance with the uninformative reward function and three settings of step size: $\alpha = 0.5$ (*a*), $\alpha = 0.1$ (*b*), and $\alpha = 0.05$ (*c*). The ordering of the curves is statistically significant at confidence level 0.005 between episodes 6100 and 22100 (*b*), and 10000 and 40000 (*c*).

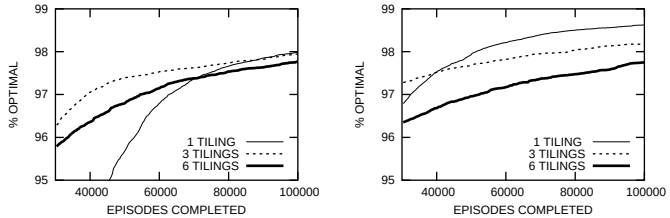


Figure 6: Long-term performance (episodes 50000–100000) with the uninformative (*a*) and informative (*b*) reward functions. The ordering of the curves in *b* is statistically significant at confidence level 0.005 staring at episode 55000.

3.3 Eventual degradation of performance due to generalization

Sections 3.1 and 3.2 demonstrate that generalization can improve performance while the policy is undergoing initial development or early refinement. Fig. 6, on the other hand, shows that in the final count generalization proves detrimental. Fig. 6 was obtained using the uninformative (*a*) and informative (*b*) reward functions. In the former case, the more challenging nature of the task favors the use of generalization for a longer time.

4 Interpretation of empirical results

We start by introducing a formal abstraction of the problem. Consider a generic domain with a continuous action variable (any additional state/action variables can be continuous or discrete). For analysis purposes, we categorize state-action pairs as *overestimated*, *underestimated*, and *correctly estimated* with respect to the backup procedure used and the approximate value function $\hat{Q}(s,a)$ for states and actions. In Q-learning, an overestimated state-action pair is characterized by

$$\hat{Q}(s,a) > R(s,a) + \gamma \max_{a' \in \mathcal{A}} \hat{Q}(s',a'),$$

where s' is the successor state. Underestimated and correctly estimated state-action pairs are defined by replacing the "greater than" sign in the above equation with "less than" and "equals" signs, respectively. Finally, we define a state-action pair (s,a) to be *desirable* if a is a near-optimal action in state s, i.e.,

$$|Q^*(s,a) - \max_{a' \in \mathcal{A}} Q^*(s,a')| < \delta,$$

where Q^* is the optimal value function and $\delta > 0$ is a small constant. *Undesirable* state-action pairs are defined symmetrically.

The effect of generalization on correctly estimated state-action pairs is nonexistent or negligible since backups in such cases generate zero expected error. The effect of general-

ization on overestimated and underestimated state-action pairs, on the other hand, is significant. In what follows, we analyze these two cases separately. We assume the exploration/exploitation trade-off is addressed using Boltzmann (softmax) action selection, an ϵ-greedy policy, or any other method in which the greedy action $\hat{a}^* = \arg\max_{a \in \mathcal{A}} \hat{Q}(s, a)$ is selected in state s with the greatest probability and the probabilities of selection of the other actions are nondecreasing in their value estimates. We refer to the region to which a value update of a state-action pair (s, a) is generalized as the *vicinity of* (s, a).

4.1 Generalization on overestimated vs. underestimated state-actions pairs

Generalizing the value update of an overestimated state-action pair (s, a) to nearby state-action pairs will decrease their value estimates and thus reduce the likelihood of selection of the corresponding actions in their respective states. If (s, a) and state-action pairs in its vicinity are undesirable, this generalized update is beneficial *regardless* of whether these state-action pairs are also overestimated. If, on the other hand, some state-action pairs in the vicinity of (s, a) are desirable, generalization is harmful if they are not overestimated. In this latter case, generalization will excessively lower the probability of selection of certain good actions.

Similarly, generalization on an underestimated state-action pair (s, a) is helpful if the state-action pairs in its vicinity are desirable, and may be harmful if there are undesirable pairs with correct or excessive estimates. However, there is an additional benefit to generalizing on desirable underestimated state-action pairs. Typical domains have continuous value functions. In this case, generalization ensures that the nearby state-action pairs also have favorable estimates *even if they are rarely tried*. Unless a large step-size is used, generalization thus accelerates the adoption of better actions in the vicinity of (s, a) as greedy choices. By contrast, a non-generalizing learner will require more exploratory visits to the vicinity of (s, a) to build up these actions' value estimates.

4.2 Application to the empirical results

Generalization improves early performance in Section 3.1 when used with the more informative reward function because the algorithm can more rapidly learn clusters of actions that lead to a fall off the cliff. When such a catastrophic event occurs and a heavy penalty is received (-100), the learner generalizes the outcome to neighboring p values, thus requiring less time to identify directions of travel to avoid *for any value of* p. A non-generalizing learner, on the other hand, needs to visit every p value within resolution to rule out a poor choice of direction. This is an example of the beneficial effects of generalization on overestimated state-action pairs. The uninformative reward function, on the other hand, does not communicate the undesirability of falling off the cliff and leads to no performance improvement with generalization.

The small step sizes used in Section 3.2 require a substantial amount of exploratory activity to build up value estimates for better p choices in the vicinity of an already established one—unless generalization across tiles is used, yielding elevated value estimates for those p choices even if they are rarely tried. As a result, generalization improves performance for small step sizes, a benefit of generalization on underestimated state-action pairs. The use of the uninformative reward function ensures that the performance differences are not due to the expedited mastery of cliff avoidance with generalization.

Once the value estimates become sufficiently accurate (with the optimum actions adopted as greedy choices and the catastrophic actions assigned low values), generalization cannot further improve performance. The negative effects of generalization are, on the other hand, still at work. The empirical results in Section 3.3 confirm that generalization is detrimental at the final stages. As expected, the observed performance degradation is monotonic in the breadth of generalization.

5 Extensions

We have confirmed that our empirical findings in Section 3 scale with map size. As an example, Fig. 7 shows the early and late performance curves using the informative reward function and a 32×8 grid world. This new map is 6.4 times larger than that of Fig. 2 but structurally similar to it.

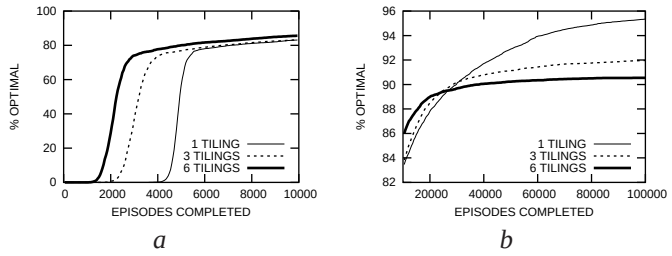


Figure 7: Performance in a 32×8 grid world: episodes 0–10000 (*a*) and 10000–100000 (*b*). The ordering of the curves is statistically significant at confidence level 0.005 between episodes 2000 and 5600 (*a*) and starting at episode 40000 (*b*).

The empirical results in Section 3 indicate that broad generalization is helpful at the early stages of learning and detrimental in the final count, suggesting that on-line adjustment of generalization breadth could yield the optimal approach. To this end, we implemented an adaptive algorithm as follows. For every state-action pair (s, a), our method maintains a *reliability index* $\rho(s, a)$ that expresses the learner's confidence in $\hat{Q}(s, a)$, ranging from 0 (unreliable) to 1 (reliable). The reliability indices (initialized to 0) are stored in a tiling organization identical to that for the Q-values themselves. Backups of $\hat{Q}(s, a)$ that yield a large error lower the reliability indices for (s, a) and nearby state-action pairs; backups that result in a small error increase those reliability indices. When performing a backup, our algorithm selects the largest allowable breadth of generalization such that the variable space covered has an average reliability index of less that 1/2. This heuristic encourages broad generalization when the value estimates are rapidly changing and discourages generalization when they are near convergence. Fig. 8*a* illustrates the progress of generalization breadth on a typical run. Figs. 8*b* and 8*c* demonstrate that this approach is superior to fixed settings of generalization breadth between episode numbers 450–49000, which arguably covers any reasonable allotment of training time in this domain. These results demonstrate that varying generalization breadth is generally preferable to a fixed setting.

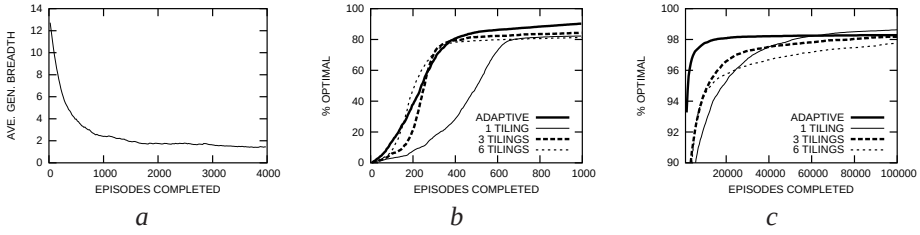


Figure 8: Adaptive method in the 10×4 grid world: per-episode average generalization breadth, smoothed (*a*); and comparative performance, episodes 0–1000 (*a*) and 1000–100000 (*b*). At confidence level 0.005, the adaptive method is superior between episodes 450 and 49000.

6 Conclusions

This paper introduces a simulated domain for the controlled study of a promising method for continuous-action RL and reports empirical results on its performance in conjunction

with Q-learning. Our experimental findings contribute a deeper understanding of the workings of tile coding, provide guidance to parameter choices, and point out an improvement on this method. Specifically, broader generalization proves beneficial in the short-term if the action space includes catastrophic choices and the reward function effectively communicates the undesirability of such actions. Broader generalization also dominates at the early stages of policy refinement by facilitating hill climbing in the action space. In the long term, however, generalization proves detrimental to the control task. If fixed generalization breadth is used, its setting should therefore be chosen based in part on the learning time allotment. Our findings suggest that varying generalization breadth on-line yields the optimal performance. We outline one algorithm for doing this and empirically demonstrate its effectiveness in our testbed domain.

7 Acknowledgments

This research was supported in part by NSF CAREER award IIS-0237699 and an MCD fellowship.

Appendix: Proof of Theorem 1

Proof: (Sketch.) The theorem can be proven by establishing that any function representable with a t-tiling organization is also representable with a single-tiling organization, and vice versa. The former claim is shown by projecting the t-tiling organization onto the single-tiling organization and weighting the tiles of the single-tiling organization by the sum of the corresponding tile weights of the t-tiling organization. The latter claim is shown by assigning random weights to the leftmost $t-1$ tiles of the t-tiling organization (one in each of the first $t-1$ tilings) and weighting the leftmost tile in the remaining tiling such that the sum of the t tile weights equals the weight of the first tile in the single-tiling organization; the latter weighting operation is repeated iteratively, moving at each step one tile to the right in the t-tiling organization. □

References

[1] Chris Gaskett, David Wettergreen, and Alexander Zelinsky. Q-learning in continuous state and action spaces. In *Australian Joint Conference on Artificial Intelligence*, pages 417–428, 1999.

[2] H. Gross, V. Stephan, and M. Krabbes. A neural field approach to topological reinforcement learning in continuous action spaces. In *IEEE World Congress on Computational Intelligence, WCCI'98 and International Joint Conference on Neural Networks, IJCNN'98*, Anchorage, Alaska, 1998.

[3] C-S. Lin and H. Kim. CMAC-based adaptive critic self-learning control. In *IEEE Trans. Neural Networks*, volume 2, pages 530–533, 1991.

[4] J. Santamaría, R. Sutton, and A. Ram. Experiments with reinforcement learning in problems with continuous state and action spaces, 1998.

[5] P. Stone and R. S. Sutton. Scaling reinforcement learning toward RoboCup soccer. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 537–544. Morgan Kaufmann, San Francisco, CA, 2001.

[6] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[7] R. S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 8*, pages 1038–1044, Cambridge, MA, 1996. MIT Press.

[8] G. Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.

[9] C. K. Tham. *Modular On-line Function Approximation for Scaling up Reinforcement Learning*. PhD thesis, Cambridge University, Cambridge, England, 1994.

[10] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, 1989.